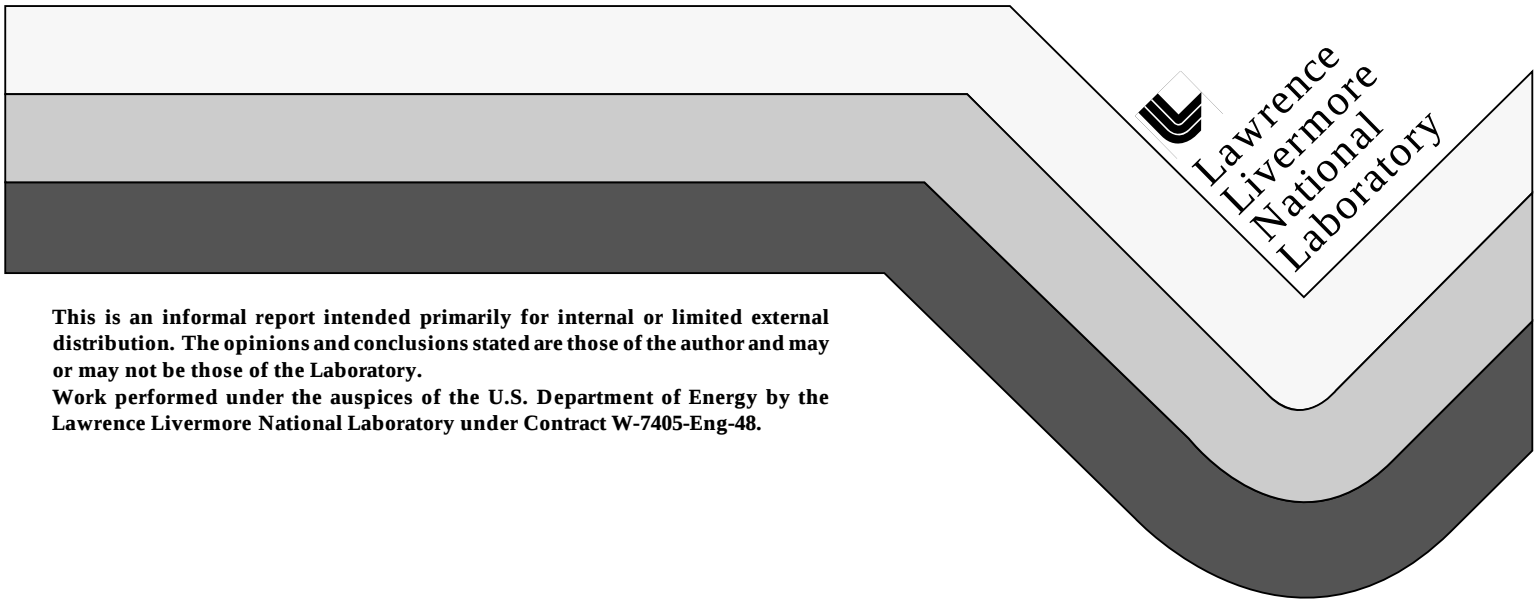
UCRL-ID-119557

# CHEMSODE: A Stiff ODE Solver for the Equations of Chemical Kinetics

**Colin J. Aro**

**January 1995**



# CHEMSODE: A Stiff ODE Solver for the Equations of Chemical Kinetics

**Colin J. Aro**
Lawrence Livermore National Laboratory
and
Dept. of Applied Science, Univ. of Calif., Davis
P.O. Box 808, L-794, Livermore, California 94550

**Subject Classifications:**

65L05: Numerical Analysis-Initial Value Problems

76AA05: Chemically Reacting Flows

**Keywords:**

Atmospheric Chemistry, Preconditioning, Stiff ODEs

# CHEMSODE: A Stiff ODE Solver for the Equations of Chemical Kinetics

**Colin J. Aro**
Lawrence Livermore National Laboratory
and
Dept. of Applied Science, Univ. of Calif., Davis
P.O. Box 808, L-794, Livermore, California 94550

**Subject Classifications:**

65L05: Numerical Analysis-Initial Value Problems

76AA05: Chemically Reacting Flows

**Keywords:**

Atmospheric Chemistry, Preconditioning, Stiff ODEs

# 1.0 Introduction

This report describes the CHEMSODE package: a collection of FORTRAN subroutines for the automatic integration of systems of ordinary differential equations arising in atmospheric chemical kinetics. These problems can be cast in the nonlinear form:

$$\frac{dy}{dt} = R(y, t) = P(y, t) - \hat{L}(y, t)\, y - \bar{L}(y, t)\, y^2 \qquad \textbf{(1)}$$

where $y(t)$ is the vector of chemical concentrations, $y^2(t)$ is the vector containing the square of each chemical concentration, $P(y, t)$ is a vector representing the "production rate" term, while $\hat{L}(y, t)$ and $\bar{L}(y, t)$ are diagonal matrices representing the "loss rate" terms. Interesting to note are the forms that these production and loss rates take. We have $P_i = \sum_q k_q y_{u_q}^{j_q} y_{v_q}^{r_q} y_{w_q}^{s_q}$, where the components of the chemical concentration vector are raised to the powers $j_q, r_q, s_q$ which are either zero or one, depending on the chemical reactions in the problem. Further, $u_q, v_q, w_q \neq i$ for each $q$, so that $P_i$ does not depend on $y_i$. The $k_q$ represent reaction rates and may be time dependent (for photochemical reactions, which are diurnally varying). Finally, $\hat{L}_{ii}$ and $\bar{L}_{ii}$ which we henceforth denote as $\hat{L}_i$ and $\bar{L}_i$, have the forms $\hat{L}_i = \sum_q k_q y_{u_q}^{j_q} y_{v_q}^{r_q}$ and $\bar{L}_i = \sum_q k_q y_{u_q}^{j_q}$ where the exponents of the chemical concentrations are again either zero or one and none of the subscripts is equal to $i$ [1, 4, 5, 6].

# 2.0 Methods and Theory

CHEMSODE uses a class of method derived from the family of implicit multistep methods called Preconditioned Time Difference methods [1]. These methods have the advantage that they are highly stable for stiff problems, while being explicitly computable and requiring no Jacobian matrix for the class of chemical kinetics equations (1).

We illustrate the basic idea behind preconditioned time differencing by considering the first order nonlinear system of ordinary differential equations:

$$\frac{dy}{dt} = f(y, t), \quad y(0) = y_0 \tag{2}$$

A k step implicit multistep formula for solving these equations is

$$y^{n+1} = \Delta t \beta_k \, f(y^{n+1}, t_{n+1}) + g^n \tag{3}$$

(with $\Delta t = stepsize$, $t_{n+1} = (n+1)\Delta t$, $y^{n+1} \cong y(t_{n+1})$, $\beta_k = const.$, and $g^n$ a known vector).

Fixed point iteration for the solution of the nonlinear system in Eq. (3) begins by defining the function

$$G(y) = \Delta t \;\; \beta_k \;\; f(y, t_{n+1}) \;\; + g^n$$

and then performing the iteration

$$(y^{n+1})^{(m+1)} = G((y^{n+1})^{(m)}), \; (y^{n+1})^{(0)} = \text{arbitrary.}$$

Convergence of the iteration to $y^{n+1}$ will occur if the initial guess $(y^{n+1})^{(0)}$ is chosen in a neighborhood of contraction of $G$ about $y^{n+1}$, [1]. If $G$ is not contractive at $y^{n+1}$ then the fixed point iteration may diverge.

The problem of diverging or slowly converging fixed point iteration schemes can often be rectified by “preconditioning”. That is, a function $F(v, u, t)$ is defined such that $F(v, v, t) = G(v, t)$; the iterates are then computed in the following fashion:

$$(y^{n+1})^{(m+1)} = F((y^{n+1})^{(m+1)}, (y^{n+1})^{(m)}, t_{n+1}), \; (y^{n+1})^{(0)} = \text{arbitrary.}$$

If this procedure were to converge, say to $y^{n+1}$, then we see that

$$y^{n+1} = F(y^{n+1}, y^{n+1}, t_{n+1}) = G(y^{n+1}, t_{n+1}) = \Delta t \beta_k f(y^{n+1}, t_{n+1}) + g^n \quad (4)$$

and the desired solution is obtained. Although each iteration requires the solution of another non-linear system of equations, this may not be difficult if the preconditioner $F$ is defined appropriately. In this report, we consider the Jacobi preconditioner $F_i(u, v, t) = \Delta t \ \beta_k \ f_i(v_1, v_2, \ldots, v_{i-1}, u_i, v_{i+1}, \ldots, v_N, t) + g_i^n$, [1].

Now, if the iteration is terminated prematurely after, say, $l$ iterations and we set $y^{n+1} = (y^{n+1})^{(l)}$, then we generate time differencing methods of the form:

1. Set $(y^{n+1})^0$ (the initial guess).

2. For $\nu = 1, 2, \ldots, l$, compute $(y^{n+1})^{(\nu)} = F((y^{n+1})^{(\nu)}, (y^{n+1})^{(\nu-1)}, t_{n+1})$

3. Set $y^{n+1} = (y^{n+1})^{(l)}$

These are the so-called Preconditioned Time Differencing methods.

## 2.1 A First Order Method

To illustrate, consider the application of the Jacobi preconditioner to the backward Euler method and use $(y^{n+1})^0 = y^n$ (known as the identity predictor) as the initial guess. Taking only a single iteration ($l = 1$), we then have:

$$y_i^{n+1} = \Delta t f_i(y_1^n, \ldots, y^n{}_{i-1}, y_i^{n+1}, y^n{}_{i+1}, \ldots, t) + y^n{}_i. \quad (5)$$

Additional iterations would yield:

$$(y_i^{n+1})^\nu = \Delta t f_i((y_1^{n+1})^{\nu-1}, \ldots, (y_i^{n+1})^{\nu-1}, (y_i^{n+1})^\nu, (y_i^{n+1})^{\nu-1}, \ldots, t) + y^n{}_i \quad (6)$$

which, when applied to Eq. (1) give us:

$$(y_i^{n+1})^\nu = y_i^n + \Delta t((P_i^{n+1})^{\nu-1} - (\hat{L}_i^{n+1})^{\nu-1}(y_i^{n+1})^\nu - (\bar{L}_i^{n+1})^{\nu-1}((y_i^{n+1})^\nu)^2) \quad (7)$$

We see that we must solve a quadratic equation to obtain $(y_i^{n+1})^{\nu}$. The two roots of this equation are a positive and a negative real number and we set $(y_i^{n+1})^{\nu}$ to be the positive root. Even with a single iteration, this method retains the first order accuracy of the backward Euler method, and seems to be unconditionally stable for the class of chemical kinetics problems [1]. Note that it is explicitly computable.

## 2.2 A Second Order Method

The implementation of a second order technique proceeds along similar lines: We start with the trapezoidal rule, and apply the Jacobi preconditioner. In this case, however, we must use at least a first order predicted value for $(y^{n+1})^{0}$ to retain the method's second order accuracy [1]. In this report, we use the first order Jacobi preconditioned method (7) to obtain this predicted value, because of its superior stability.

The algorithm and component equations for a single iteration of this method, taken from [1], are:

1. Set $(y^{n+1})^{0} = \hat{y}^{n+1}$, the computed value from the Jacobi preconditioned Backward Euler method.
2. Calculate $P^{n+1}, \hat{L}^{n+1}, \bar{L}^{n+1}$ using these values
3. Solve

$$y_i^{n+1} = \frac{\Delta t}{2}\left[ (P_i^{n+1} - \hat{L}_i^{n+1} y_i^{n+1} - \bar{L}_i^{n+1} (y_i^{n+1})^2) + (P_i^n - \hat{L}_i^n y_i^n - \bar{L}_i^n (y_i^n)^2) \right] + y_i^n \quad \textbf{(8)}$$

for $y_i^{n+1}$ using the positive root. Multiple iterations, as in (7) are implemented in the obvious way.

Error control is a significant challenge for these methods. The choices are extreme with little possibility for an "in between" compromise. In [1], we used a method of comparing the

calculation of two half steps with a single whole step. This was fairly accurate but obviously expensive. The alternative is an inaccurate estimate of a quantity that “looks like” our truncation error where we hope to err on the conservative side. We thus use the local error indicator:

$$E^{n+1} = \frac{2}{c+1}\left(cy^{n+1} - (1+c)\,y^{n} + y^{n-1}\right) \tag{9}$$

with $c = \dfrac{(t_n - t_{n-1})}{(t_{n+1} - t_n)}$. This indicator gives us an estimate of $\Delta t^2 y''(t_n) + O(\Delta t^3)$, which is a bit conservative, but was used with success in [5]. Given the values ATOL(I) and RTOL(I), the componentwise absolute and relative tolerances, we compute the value

$$max_i\left(\frac{E_i^{n+1}}{ATOL(I) + RTOL(I)\,y_i^n}\right) \equiv \left\| E^{n+1} \right\| \tag{10}$$

If this value is less than one, the step is accepted. The new stepsize is computed using the formula

$$\Delta t_{new} = \left(\frac{0.94}{\sqrt{\left\| E^{n+1} \right\|}}\right)\Delta t_{old}, \tag{11}$$

subject to prescribed minimum and maximum values (to be described later) and a maximum ratio increase.

CHEMSODE is written in the same spirit as the more general LSODE package [3], used for the automatic integration of the general first order ODE system.

# 3.0 Code

At this time, CHEMSODE uses a second order method (the trapezoidal rule) with a Jacobi preconditioner (one iteration). The package consists of the following subroutines:

### 3.0.1 SOLVER

SOLVER is the driver for the package. It takes the following arguments:

calculation of two half steps with a single whole step. This was fairly accurate but obviously expensive. The alternative is an inaccurate estimate of a quantity that “looks like” our truncation error where we hope to err on the conservative side. We thus use the local error indicator:

$$E^{n+1} = \frac{2}{c+1}\left(cy^{n+1} - (1+c)\,y^{n} + y^{n-1}\right) \tag{9}$$

with $c = \dfrac{(t_n - t_{n-1})}{(t_{n+1} - t_n)}$. This indicator gives us an estimate of $\Delta t^2 y''(t_n) + O(\Delta t^3)$, which is a bit conservative, but was used with success in [5]. Given the values ATOL(I) and RTOL(I), the componentwise absolute and relative tolerances, we compute the value

$$max_i\left(\frac{E_i^{n+1}}{ATOL(I) + RTOL(I)\,y_i^n}\right) \equiv \left\| E^{n+1} \right\| \tag{10}$$

If this value is less than one, the step is accepted. The new stepsize is computed using the formula

$$\Delta t_{new} = \left(\frac{0.94}{\sqrt{\left\| E^{n+1} \right\|}}\right)\Delta t_{old}, \tag{11}$$

subject to prescribed minimum and maximum values (to be described later) and a maximum ratio increase.

CHEMSODE is written in the same spirit as the more general LSODE package [3], used for the automatic integration of the general first order ODE system.

# 3.0 Code

At this time, CHEMSODE uses a second order method (the trapezoidal rule) with a Jacobi preconditioner (one iteration). The package consists of the following subroutines:

### 3.0.1 SOLVER

SOLVER is the driver for the package. It takes the following arguments:

- F: (EXT) Name of the subroutine for the right hand side of the ODE system (the derivative). This is supplied by the user and must be declared as EXTERNAL in the calling program, having the form

```
SUBROUTINE F(NEQ, T, Y, P, LHAT, LBAR)
INTEGER NEQ
REAL T, Y(NEQ), P(NEQ), LHAT(NEQ), LBAR(NEQ)
```

  The inputs are NEQ, T and Y (the number of ODEs, current time and chemical species concentrations). F is to set the chemical production and loss terms P(i), LHAT(i), LBAR(i).

- NEQ: (IN) The number of first order ODEs.
- Y: (INOUT) Array of components of the Y(t) (dependent variable) from Eq. (1). Y shall be dimensioned at least NEQ elements long. On initial call, it contains the species concentrations at t = T. On return, it contains the values at t = TOUT (see below).
- T: (INOUT) Value of the independent variable. Upon return, it will contain the value TOUT.
- TOUT: (IN) The next point where output is desired.
- ATOL: (IN) Array of absolute error tolerances (size NEQ).
- RTOL: (IN) Array of relative error tolerances (size NEQ).
- RWORK: (WORK) This work array must be dimensioned at least 10 * NEQ real elements long. It is also used to pass optional arguments to the solver (see below IOPT).
- IOPT: (IN) Indicates whether the user has supplied optional input. IOPT = 0 indicates no optional input. IOPT = 1 indicates that RWORK(1) contains the first stepsize to be attempted, RWORK(2) contains the maximum allowable stepsize, and RWORK(3) contains the minimum allowable stepsize. Default values recently used are RUMACH (see below), 1 second, and 1 hour respectively. A non-zero minimum is required since our “loose” method of step control has problems otherwise.
- ITASK: (IN) Indicates whether the call is a continue or a restart. ITASK = 0 indicates a restart. ITASK = 1 indicates a continue.

A flowchart for SOLVER is provided in Figure 2, and is very run-of-the-mill for this type of code. At this stage there is very little error checking and recovery, however the code has proven to be quite bullet-proof in our experiences. In adjusting the new stepsize, if two successive steps are rejected, the stepsize is automatically reduced to the minimum allowable; that is, the method is restarted.

### 3.0.2 STEP2

STEP2 takes a single discrete time step and returns an estimate of the error to the SOLVER routine. The stepsize attempted is specified by the SOLVER routine. The error estimate is based on the formula of Eq (9). This information is passed back to the SOLVER routine which calculates the quantities in Eqs. (10) and (11) and accepts or rejects the step accordingly (while adjusting the stepsize for the next try).

### 3.0.3 ADVANCE2

ADVANCE2 implements the component equations for the preconditioned trapezoidal rule assuming that the production and loss rates are set. It essentially provides for the solution of Eq (8) for $y_i^{n+1}$ assuming all other values are explicitly known (which they are). It uses a scalar newton iteration and a fixed number of iterations (two). This works well with a close enough initial guess ($y_i^n$ -- not to be confused with the initial guess for the preconditioner). Attempts at using the quadratic formula proved unsuccessful since it requires the subtraction of two positive quantities that are nearly equal.

### 3.0.4 ADVANCE1

ADVANCE1 implements the component equations for the preconditioned backward Euler method, again assuming that the production and loss rates are set. This is Eq (7) and is done in a manner identical with that in ADVANCE2.

### 3.0.5 RUMACH

RUMACH calculates the machine's unit roundoff in a machine independent manner.

**FIGURE 1. CHEMSODE subroutine calling hierarchy.**

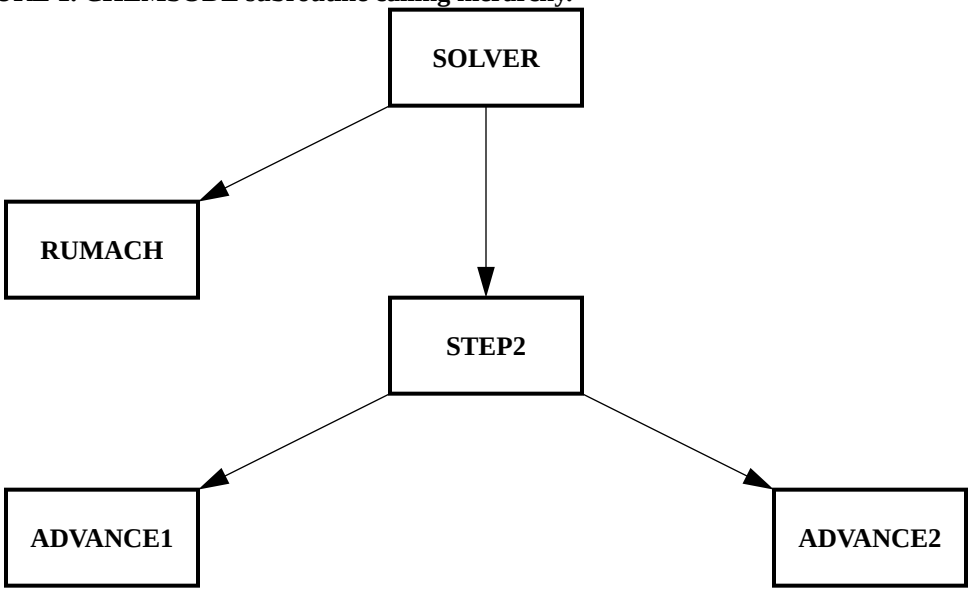
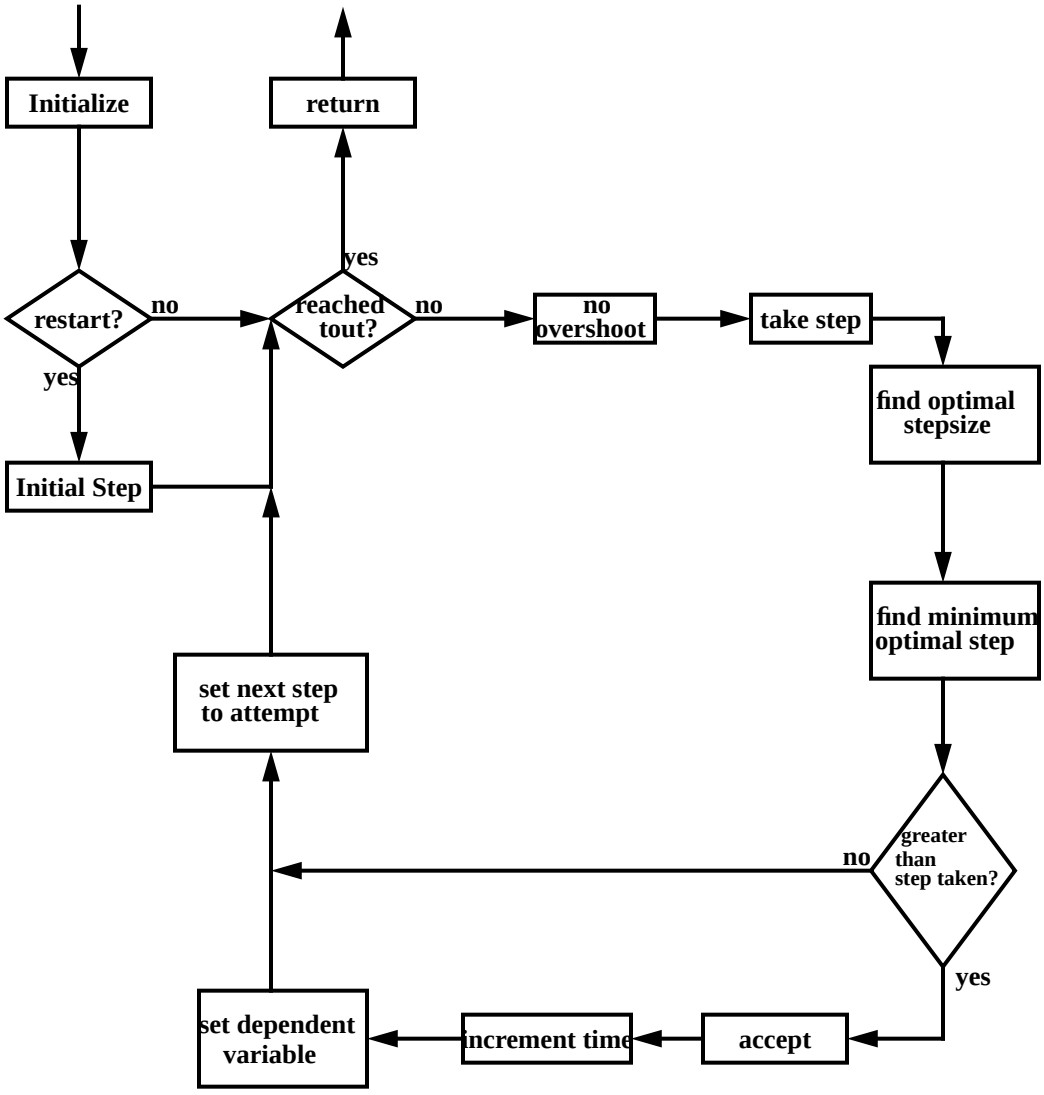


**FIGURE 2. Flowchart for the CHEMSODE algorithm. (Subroutine SOLVER)**



# 4.0 Code usage and Example Problem

The reader will find strong similarities between LSODE's calling sequence (and parameters) and CHEMSODE's. CHEMSODE's parameters shall be declared as follows:

```
INTEGER NEQ, IOPT
REAL Y(NEQ), T, TOUT, ATOL(NEQ), RTOL(NEQ)
REAL RWORK(10*NEQ)
EXTERNAL F
```

The solver is then called in the main program

```
 CALL SOLVER(NEQ, F, Y, T, TOUT, ATOL, RTOL,
1 RWORK, IOPT, ITASK)
```

## 4.1 EXAMPLE

We use a simple problem from chemical kinetics. The Chapman atmosphere has been dealt with previously [1, 2]. It can be written in a simplified form Eq. (1) for $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$, with

$$P(y,t) = \begin{bmatrix} 2k_3(t)\,y_3 + k_4(t)\,y_2 \\ k_1 y_1 y_3 \end{bmatrix} \tag{12}$$

$$\hat{L}(y,t) = \begin{bmatrix} k_1 y_3 + k_2 y_2 & 0 \\ 0 & k_2 y_1 + k_4(t) \end{bmatrix} \tag{13}$$

$$(\bar{L} = 0)$$

and the constant values

$$y_3 = 3.7 \times 10^{16}$$

$$k_1 = 1.63 \times 10^{-16}$$

$$k_2 = 4.66 \times 10^{-16}$$

and

$$k_i = \begin{cases} exp\left[\dfrac{-a_i}{\sin\omega t}\right], \sin\omega t > 0 \\ 0, \sin\omega t \le 0 \end{cases}, i = 3, 4 \quad (14)$$

with

$$a_3 = 22.62, a_4 = 7.601, \omega = \frac{\pi}{43200}$$

The constant 43200 is just twelve hours in seconds, so that the diurnally varying rates ($k_3$ and $k_4$) have 24 hour periods. Initial conditions are $y_1(0) = 10^6, y_2(0) = 10^{12}$. The Chapman mechanism has important characteristics shared by the larger problems in atmospheric chemistry in that:

- The Jacobian of the differential system is non-constant.
- The diurnal effect is present.
- The oscillations are fast (in comparison to the scale of integration),

cf. [1, 2]. This problem can be coded:

```
PROGRAM TEST

INTEGER NEQ, IOPT
REAL Y(2), T, TOUT, ATOL(2), RTOL(2)
REAL RWORK(20)
EXTERNAL FEX

IOPT = 0
ITASK = 0

NEQ = 2
T = 0.0
TOUT = 1.0 * 86400.0
ATOL(1) = 1.0E-2
ATOL(2) = 1.0E-2
RTOL(1) = 1.0000E-4
RTOL(2) = 1.0000E-4
Y(1) = 1.0E6
Y(2) = 1.0E12
```

```
 CALL SOLVER(NEQ, FEX, Y, T, TOUT, ATOL, RTOL,
1RWORK, IOPT, ITASK)

 PRINT *, 'VALUES AT TIME ', T, ' ARE ', Y

 END



 SUBROUTINE FEX(NEQ, T, Y, P, LHAT, LBAR)

 INTEGER NEQ
 REAL T, Y(*), P(*), LHAT(*), LBAR(*)

 REAL K1, K2, Y3
 REAL K3, K4
 REAL K
 PARAMETER (K1 = 1.63E-16)
 PARAMETER (K2 = 4.66E-16)
 PARAMETER (Y3 = 3.7E16)


 P(1) = 2.0 * K3(T) * Y3 + K4(T) * Y(2)
 P(2) = K1 * Y(1) * Y3

 LHAT(1) = K1 * Y3 + K2 * Y(2)
 LHAT(2) = K2 * Y(1) + K4(T)

 LBAR(1) = 0.0
 LBAR(2) = 0.0


 RETURN
 END


 REAL FUNCTION K3(T)

 REAL T

 REAL PI, OMEGA, A3
 PARAMETER (PI = 3.14159265358979323846264338)
```

```
PARAMETER (OMEGA = PI/43200.0)
PARAMETER (A3 = 22.62)
REAL TEMP

TEMP = SIN(T * OMEGA)

IF (TEMP .GT. 0.0) THEN
 K3 = EXP(-A3 / TEMP)
ELSE
 K3 = 0.0
END IF

RETURN
END




REAL FUNCTION K4(T)

REAL T

REAL  PI, OMEGA, A4
PARAMETER (PI = 3.14159265358979323846264338)
PARAMETER (OMEGA = PI/43200.0)
PARAMETER (A4 = 7.601)
REAL TEMP

TEMP = SIN(T * OMEGA)

IF (TEMP .GT. 0.0) THEN
K4 = EXP(-A4 / TEMP)
ELSE
 K4 = 0.0
END IF

RETURN

END
```

## 5.0 References


1. C. J. Aro and G. H. Rodrigue, “Preconditioned Time Differencing for Stiff ODEs in Diurnal Atmospheric Kinetics”, Report UCRL-JC-118092, LLNL, Livermore, CA, 1994 (unpublished)

2. G. D. Byrne and A. C. Hindmarsh, “Stiff ODE Solvers: A Review of Current and Coming Attractions”, *Journal of Computational Physics*, Vol. 70, No. 1, Academic Press, Inc., 1987, pp. 1-62

3. K. Radhakrishnan and A. C. Hindmarsh, “Description and Use of LSODE, the Livermore Solver for Ordinary Differential Equations”, Report UCRL-ID-113855, LLNL, Livermore, CA, 1993

4. J.I. Steinfeld, J.S. Francisco, and W.L. Hase, *Chemical Kinetics and Dynamics*, Prentice Hall, Englewood Cliffs, New Jersey (1989)

5. J. G. Verwer. “Gauss-Seidel Iteration for Stiff ODEs from Chemical Kinetics”. Report NM-R9315, CWI, Amsterdam, 1993 (in press SIAM *J. Sci. Comput.*)

6. J.G. Verwer, M. van Loon, “An Evaluation of Explicit Pseudo-Steady State Approximation Schemes for Stiff ODE Systems from Chemical Kinetics”. CWI Report NM-R9312, Centre for Mathematics and Computer Science, Amsterdam, 1993 (in press *J. Comput. Phys*.)

7. D. J. Wuebbles, J. S. Tamaresis, and K. O. Patten, “Quantified Estimates of Total, GWPs for Greenhouse Gases Taking into Account Tropospheric Chemistry”, Report UCRL-ID-115850, LLNL, Livermore, CA, 1993. (unpublished)


## 6.0 Appendix: The Box Model

CHEMSODE is currently being applied to a set of chemical kinetic equations taken from the LLNL two dimensional chemical-radiative-transport model of the earth’s atmosphere [7]. The “Box Model” has been developed to isolate these equations from the other unrelated processes taking place in the full models (e.g. advection, diffusion). It uses the same

chemistry and associated radiative transfer calculations as the full model. The attempt is to isolate a single zone from the full model in order to more accurately test the merits of various chemical kinetics solvers.

Initial tests are being run with a set of twenty chemical species listed in the following tables. Photolytic and thermal reactions, along with the two species assumed to be in equilibrium are also given.

**TABLE I. Chemically reactive species used in the box model.**

| | | | | | |
|---|---|---|---|---|---|
| $O_3$ | $N_2O$ | $NO$ | $NO_2$ | $NO_3$ | $N_2O_5$ |
| $HONO$ | $HNO_3$ | $HO_2NO_2$ | $H_2O$ | $OH$ | $HO_2$ |
| $H_2O_2$ | $H_2$ | $CH_4$ | $CH_3O_2$ | $CH_3OOH$ | $CH_2O$ |
| $CO$ | $CH_3O_2NO_2$ | | | | |

**TABLE II. Chemical species assumed to be in steady state for the box model.**

| | |
|---|---|
| $O(^1D)$ | $O$ |

**TABLE III. Thermal reactions in the Box Model**

| | | |
|---|---|---|
| $O + O_2$ | $\rightarrow$ | $O_3$ |
| $O + O_3$ | $\rightarrow$ | $2O_2$ |
| $O(^1D) + N_2$ | $\rightarrow$ | $O + N_2$ |
| $O(^1D) + O_2$ | $\rightarrow$ | $O + O_2$ |
| $O(^1D) + O_3$ | $\rightarrow$ | $2O_2$ |
| $O(^1D) + O_3$ | $\rightarrow$ | $O_2 + 2O$ |
| $H_2O + O(^1D)$ | $\rightarrow$ | $2OH$ |
| $O(^1D) + H_2$ | $\rightarrow$ | $OH + HO_2$ |
| $N_2O + O(^1D)$ | $\rightarrow$ | $N_2 + O_2$ |
| $N_2O + O(^1D)$ | $\rightarrow$ | $2NO$ |

**TABLE III. Thermal reactions in the Box Model**

| | | |
|---|---|---|
| $CH_4 + O(^1D)$ | $\rightarrow$ | $CH_2O + H_2$ |
| $CH_4 + O(^1D)$ | $\rightarrow$ | $CH_3O_2 + OH$ |
| $H_2 + OH$ | $\rightarrow$ | $H_2O + HO_2$ |
| $OH + O_3$ | $\rightarrow$ | $HO_2 + O_2$ |
| $OH + O$ | $\rightarrow$ | $O_2 + HO_2$ |
| $HO_2 + O$ | $\rightarrow$ | $OH + O_2$ |
| $HO_2 + O_3$ | $\rightarrow$ | $OH + 2O_2$ |
| $HO_2 + OH$ | $\rightarrow$ | $H_2O + O_2$ |
| $HO_2 + HO_2$ | $\rightarrow$ | $H_2O_2 + O_2$ |
| $2HO_2 + H_2O$ | $\rightarrow$ | $H_2O_2 + O_2 + H_2O$ |
| $H_2O_2 + OH$ | $\rightarrow$ | $H_2O + HO_2$ |
| $NO + O_3$ | $\rightarrow$ | $NO_2 + O_2$ |
| $NO + OH$ | $\rightarrow$ | $HONO$ |
| $NO + HO_2$ | $\rightarrow$ | $NO_2 + OH$ |
| $NO_2 + O$ | $\rightarrow$ | $NO + O_2$ |
| $NO_2 + O_3$ | $\rightarrow$ | $NO_3 + O_2$ |
| $NO_2 + HO_2$ | $\rightarrow$ | $HO_2NO_2$ |
| $NO_3 + NO_2$ | $\rightarrow$ | $N_2O_5$ |
| $N_2O_5$ | $\rightarrow$ | $NO_3 + NO_2$ |
| $NO_2 + OH$ | $\rightarrow$ | $HNO_3$ |
| $HONO + OH$ | $\rightarrow$ | $H_2O + NO_2$ |
| $HNO_3 + OH$ | $\rightarrow$ | $H_2O + NO + O_2$ |
| $HNO_3 + OH$ | $\rightarrow$ | $H_2O + NO_2 + O$ |
| $HO_2NO_2$ | $\rightarrow$ | $HO_2 + NO_2$ |
| $HO_2NO_2 + OH$ | $\rightarrow$ | $H_2O + NO_2 + O_2$ |
| $CO + OH$ | $\rightarrow$ | $HO_2$ |
| $CH_4 + OH$ | $\rightarrow$ | $CH_3O_2 + H_2O$ |
| $CH_2O + OH$ | $\rightarrow$ | $H_2O + HO_2 + CO$ |
| $CH_3O_2 + O$ | $\rightarrow$ | $CH_2O + HO_2$ |

**TABLE III. Thermal reactions in the Box Model**

| | | |
|---|---|---|
| $CH_2O + O$ | $\rightarrow$ | $HO_2 + OH + CO$ |
| $CH_3O_2 + HO_2$ | $\rightarrow$ | $CH_3OOH + O_2$ |
| $CH_3O_2 + CH_3O_2$ | $\rightarrow$ | $2CH_2O + 1.4HO_2$ |
| $CH_3O_2 + NO$ | $\rightarrow$ | $HO_2 + CH_2O + NO_2$ |
| $CH_3O_2 + NO_2$ | $\rightarrow$ | $CH_3O_2NO_2$ |
| $CH_3O_2NO_2$ | $\rightarrow$ | $CH_3O_2 + NO_2$ |
| $CH_3OOH + OH$ | $\rightarrow$ | $CH_2O + H_2O + OH$ |
| $CH_3OOH + OH$ | $\rightarrow$ | $CH_3O_2 + H_2O$ |

**TABLE IV. Photolytic Reactions in the Box Model**

| | | |
|---|---|---|
| $O_2 + h\nu$ | $\rightarrow$ | $2O$ |
| $O_3 + h\nu$ | $\rightarrow$ | $O + O_2$ |
| $O_3 + h\nu$ | $\rightarrow$ | $O\,(^1D) + O_2$ |
| $H_2O_2 + h\nu$ | $\rightarrow$ | $2OH$ |
| $NO_2 + h\nu$ | $\rightarrow$ | $NO + O$ |
| $N_2O + h\nu$ | $\rightarrow$ | $N_2 + O\,(^1D)$ |
| $NO_3 + h\nu$ | $\rightarrow$ | $NO_2 + O$ |
| $NO_3 + h\nu$ | $\rightarrow$ | $NO + O_2$ |
| $N_2O_5 + h\nu$ | $\rightarrow$ | $NO_2 + NO + O_2$ |
| $N_2O_5 + h\nu$ | $\rightarrow$ | $2NO_2 + O$ |
| $N_2O_5 + h\nu$ | $\rightarrow$ | $NO_2 + NO + 2O$ |
| $N_2O_5 + h\nu$ | $\rightarrow$ | $2NO + O_2 + O$ |
| $HONO + h\nu$ | $\rightarrow$ | $OH + NO$ |
| $HNO_3 + h\nu$ | $\rightarrow$ | $OH + NO_2$ |
| $HO_2NO_2 + h\nu$ | $\rightarrow$ | $OH + NO + O_2$ |
| $HO_2NO_2 + h\nu$ | $\rightarrow$ | $OH + NO_2 + O$ |
| $HO_2NO_2 + h\nu$ | $\rightarrow$ | $HO_2 + NO_2$ |
| $CH_2O + h\nu$ | $\rightarrow$ | $CO + H_2$ |
| $CH_2O + h\nu$ | $\rightarrow$ | $2HO_2 + CO$ |

**TABLE IV. Photolytic Reactions in the Box Model**

| | | |
|---|---|---|
| $CH_3OOH + h\nu$ | $\rightarrow$ | $CH_2O + HO_2 + OH$ |
| $CH_3O_2NO_2 + h\nu$ | $\rightarrow$ | $CH_3O_2 + NO_2$ |

*Technical Information Department* • **Lawrence Livermore National Laboratory**
**University of California • Livermore, California 94551**